Java™ SE and EE Platforms: The High Order Bits

Craig R. McClanahan
Senior Staff Engineer
Sun Microsystems, Inc.

\[ \text{Web} = \sum_{n=1}^{\infty} \text{web}_n \]
Agenda

• Java Standard Edition 6
  > Qualities
  > Features
• Java Enterprise Edition 5
  > How and Now
• Future Directions
• Summary
Java Standard Edition 6
SE Platform Schedule

• Major Releases:
  > Java SE 5 – 3Q2004
  > Java SE 6 – 4Q2006
  > Java SE 7 – 3-4Q2008
• Plus bugfix updates every 8-16 weeks
• No interim releases like 5.1 or 6.1
SE Platform Development Model

• https://jdk6.dev.java.net

• Goal: Become more open
  > With both sources and binaries

• Method: Deliver weekly snapshot releases
  > Quick turnaround on bug fixes
  > Immediate feedback on features
  > Changing our whole approach on betas

• Enables community contributions
  > From bug fixes to features
What Makes The SE Platform Great?

• Systemic Properties:
  > Compatibility
  > Stability
  > Quality

• Performance Improvements

• Monitoring and Management

• Continued Flow of New Features
Systemic Properties

• Improve quality, but keep old programs working:
  > Single most important focus for the team

• Many important new process initiatives:
  > Continual internal process improvements
  > Weekly binary snapshots
  > “Regression Challenge”
  > “Crack The Verifier Challenge”

• We tend to be paranoid about fixing bugs:
  > One person's bugfix is another person's regression
  > But we've still been able to fix many bugs!
Performance

- Java SE 6 continues to focus on improved performance in many areas:
  - On both the client and the server
  - Upgrades to core JVM execution
  - Garbage collection scaling and parallelism
  - Library tuning
  - Upgrades to Java 2D graphics rendering
  - Improvements in application startup
Performance – Example

• Time to repaint exposed region in a large Swing application – smaller is better
Monitoring and Management

• Key value add here in Java SE 5:
  > Embedded Java Management Extensions (JMX) support

• Further improvements in Java SE 6:
  > Even more JVM level diagnostics
    > Example: Out of memory handling
  > Improved Solaris DTrace integration
  > “Attach on demand” monitoring
  > Jconsole upgrades
New Features – Scripting

• A new framework for scripting engines (JSR-223):
  > Allows plugging in new languages ... 
  > And call into the Java platform 

• Sun is co-packaging JavaScript language support:
  > Using the Rhino engine from Mozilla 

• Scripting project on Java.Net supports many more:
  > Groovy, Jelly, Phyton, Ruby, Scheme, ... 
  > Plus links to BeanShell and PHP plugins 

> https://scripting.dev.java.net
New Features – Desktop

• Improved look-and-feel support:
  > Significant updates on Windows and Gnome
  > New support for Windows Vista
  > Antialiased LCD fonts

• Swing and Java 2D performance improvements

• Assorted API upgrades:
  > SwingWorker, JTable sorting, GroupLayout

• Desktop integration:
  > Icons in system tray, splash screens
  > Ability to launch native tools
Windows Vista Support

• Windows Vista is a key target platform

• Java SE 6 will include core Vista support:
  > Including IE7 plugin, Vista look-and-feel
  > All available when Vista ships
  > Call To Action – help test your applets and applications on Vista and Java SE 6

• Backport some Vista support to 1.4.2 and 5.0:
  > But with older UI, more limited plugin support
New Features – JAX-WS

• Goal – heterogeneous distributed programming
  > Successor to CORBA, RMI, DCOM, ...
  > Easy to use
  > Rich data types
  > Entire industry on board, including Microsoft

• Java SE 6 delivers a full JAX-WS web service client
  > Plus a lightweight server for asynchronous callbacks
  > Java EE 5 provides richer server support
Java SE 6 – The Rest Of The Story

• Why is Java SE 6 great?
  > Systemic improvements
  > Many more new features

• Call To Action:
  > Learn more about Java SE 6 capabilities:
    > http://java.sun.com/javase
  > Test your applets and applications against SE 6:
    > https://jdk6.dev.java.net
  > Start planning how you can leverage this new platform, when it releases later this quarter
Java Enterprise Edition 5
Java EE 5 – Most Important Things

• It is done!
  > Released May 2006

• It has industry support:
  > Sun has already released a compliant product
  > Several other vendors are very close

• It accomplished the major goal set at the beginning:
  > Make it easier to develop Java EE applications
  > Especially when first getting started with Java EE
How Did We Make It Easier?

• Declarative programming:
  > Original model based on deployment descriptors
  > Now, language annotations provide most info
    > But you can override with descriptors if needed

• Remove requirements:
  > Originally based on implementing platform APIs
  > Now, plain old Java objects (POJOs)
  > Plus, better defaults

• More powerful frameworks:
  > Less work for you to do
How Did We Make It Easier?

• Declarative programming:
  > Original model based on deployment descriptors
  > Now, language annotations provide most info
    > But you can override with descriptors if needed

• Remove requirements:
  > Originally based on implementing platform APIs
  > Now, plain old Java objects (POJOs)
  > Plus, better defaults

• More powerful frameworks:
  > Less work for you to do
Annotations in Java EE 5

• Defining and using web services
• Mapping Java classes $\leftrightarrow$ XML
• Greatly simplify Enterprise JavaBean (EJB) development
• Map Java classes to databases
• Specify external dependencies
• Reduce/eliminate need for deployment descriptors
• We are just starting to scratch the surface of what is possible
Major Features in Java EE 5

- Simplified web services support:
  > Including support for more web service standards
- Dependency injection
- Greatly simplified EJB development
- New Java Persistence API (JPA)
- JavaServer Faces (JSF) for easy web development

- And, backwards compatible with J2EE 1.4
Example – Creating A Web Service

Here is a simple business logic bean:

```java
package endpoint;

public class Hello {

    public String sayHello(String name) {
        return "Hello " + name;
    }
}
```
Example – Creating A Web Service

• Let's turn it into a web service:

```java
package endpoint;
import javax.jws.WebService
@WebService
public class Hello {

    public String sayHello(String name) {
        return "Hello " + name;
    }
}
```
Example – Using A Web Service

```java
package client;
import endpoint.Hello;
import javax.xml.ws.WebServiceRef;
public class Client {
    @WebServiceRef(Hello.class)
    private static Hello svc;

    public static void main(String args) {
        System.out.println(svc.sayHello(args[0]));
    }
}
```
Web Services in Java EE 5

• JAX-WS and JAXB under the covers:
  > But you do not need to understand the details
• Supports the latest W3C standards:
  > SOAP/1.2, MTOM/XOP, XML Schema 1.0
• Plus the latest WS-I standards:
  > Basic Profile 1.1, Attachment Profile 1.0
• Starting to support WS-* specifications
  > WS-Security
• More to follow in future versions
Enterprise JavaBeans 3.0

• Dramatic simplification for all bean types
• POJO based EJBs
• More defaults, less configuration
• Dependency injection
• Interceptors
package endpoint;
import javax.jws.WebService;
import javax.ejb.Stateless;
@WebService
@Stateless
public class Hello {
    public String sayHello(String name) {
        return "Hello " + name;
    }
}

Java Persistence API

• Single persistence API for Java SE and Java EE
• Developed by EJB 3.0 Expert Group
  > Builds on years of experience with existing technologies and products
• Much simpler than EJB 2.1 CMP
• At least three implementations so far (open source):
  > Oracle – TopLink Essentials (used in Glassfish)
  > JBoss – Hibernate
  > BEA – Kodo / OpenJPA
package entities;
import javax.persistence.*
@Entity
public class Person {

@Id private int personId;
// getPersonId(), setPersonId(), ...

private String name;
// getName(), setName(), ...
}

Dependency Injection

• Example of “inversion of control” principles
  > Avoids the need to use JNDI for resource lookups

• Container injects resources ...
  > DataSource, EJB/WS refs, persistence units, UserTransaction, etc.

• Into application provided classes ...
  > Fields or methods, protected or public

• Where the container creates instances
  > EJBs, servlets, JSF managed beans, web service endpoints, handlers, interceptors, app clients
Dependency Injection – Example

• My EJB needs a JDBC DataSource:

```java
@Stateless

public class Hello {

    // Injected by the container
    // No deployment descriptor required!
    @Resource(name="jdbc/Customers")
    private DataSource ds;

    ...

}
```
JavaServer Faces 1.2

- Standard web framework for Java EE:
  > Component model for visual components
  > Dependency injection into managed beans
  > Expression language shared with JSP 2.1

- Large market of JSF components:
  > Hundreds of components from multiple vendors
  > Commercial and open source

- Easy way to integrate AJAX functionality into your applications
Java EE 5 – New Specifications

- JSP Standard Tag Library (JSTL) (JSR-52)
- Streaming Parser for XML (StaX) (JSR-173)
- Web Services Metadata (JSR-181)
- Java Persistence API (JPA) (JSR-220)
- Java API for XML Binding (JAXB) (JSR-222)
- Java APIs for Web Services (JAX-WS) (JSR-224)
- Common Annotations (JSR-250)
- JavaServer Faces (JSF) (JSR-252)
Java EE 5 – Updated Specifications

• Enterprise JavaBeans 3.0 (EJB) (JSR-220)
• JavaServer Pages 2.1 (JSP) (JSR-245)
• Minor Updates:
  > Enterprise Web Services (JSR-109)
  > Servlets
  > JavaMail, Java Activation Framework
  > Management, Deployment
  > JACC, SAAJ, JTA, …
So, How Much Easier Is It?

- **Adventure Builder:**
  > J2EE 1.4 – 67 classes, 3284 lines of code
  > Java EE 5 – 43 classes, 2777 lines of code
  > **36%** fewer classes to manage!

- **Roster App:**
  > J2EE 1.4 – 17 classes, 987 lines of code
  > Java EE 5 – 7 classes, 716 lines of code
  > J2EE 1.4 – 9 XML files, 792 lines
  > Java EE 5 – 1 XML file, 5 lines
  > **58%** fewer classes, **89%** fewer XML files!
And, It Is Open Sourced

- Project Glassfish at Java.Net:
  - [https://glassfish.dev.java.net](https://glassfish.dev.java.net)
  - CDDL license
  - Basis for Sun Java System Application Server 9
- **The Aquarium** (Aggregation of Blogs):
- Contains all the elements of Java EE 5
Java EE 5 – Summary

• Development with Java EE 5 is much easier

• Call To Action – Download the SDK and docs
  > http://java.sun.com/javaee

• Call To Action – Get involved in the Glassfish community
  > https://glassfish.dev.java.net

• Call To Action – Give us feedback
  > javaee-spec-feedback@sun.com
  > http://forum.java.sun.com
Future Directions
So What Will Happen Next

• The next chapters of the Java SE and Java EE platform stories have yet to be written
• What will happen next?
  > Here are some things we are looking at
  > **Call To Action** – give us your ideas also!
• Here are some ideas we are looking at ...
Java SE 7 – Language Updates

- We are cautious about changing the Java language
  - The current language works really well
  - But some thoughtful change seems good
- Direct support for XML
  - Under active investigation
  - Allows XML inlining plus simple XPath queries
- “Super Packages” (JSR-292)
  - Enhance package isolation support
  - Complements runtime module work (JSR-277)
- Probably some others as well
Java SE 7 – Other Ideas

• Desktop improvements:
  > More Java 2D graphics acceleration
  > Swing “Beans Binding” API (JSR-295)
  > Swing application framework (JSR-296)

• Java module system (JSR-277):
  > Better packaging support
  > Includes a package repository

• BeanShell (JSR-274) integration

• Many more candidates to consider
Growing The Java Platform

• We want to grow the platform in several directions
Different Uses / Different Languages

• Java is the “gold standard” for enterprise computing
  > Great for robust, long lived, maintainable apps

• But diversity is good!
  > Lightweight apps may not need static type checks
  > Fast-changing presentation tier can use different styles of development
    > But still need to call into Java business logic

• Advice – Support mixed-mode development
  > Different styles for different needs, on the shared Java platform
Dynamic Languages

• Many dynamic languages run on the JVM today
  > Ruby, Python, Groovy, many others

• New bytecode will accelerate dynamic languages:
  > Supporting Dynamically Typed Languages (JSR-292)
  > JVM designed originally to support Java
  > Dynamic languages can benefit from flexibility in method dispatch

• First bytecode not used by Java language itself
Visual Basic for Java

- Goal – enable VB developers to use Java platform
- Project *Semplice* – Support VB language on (J)VM
  - Compile from VB source to bytecodes
  - VB source can call into Java platform APIs
- *Not* intended to emulate all of the Windows Platform APIs that VB programs use
  - Instead, allows reuse of VB development skills
Enterprise Edition Future Directions

• Grow Upwards:
  > Composite applications (JBI, SCA)
  > Portlets
  > High availability, clustering, other features

• Grow Downwards:
  > Scripting (client side and server side)
  > Web/application hosting, WebDAV

• Grow Sideways:
  > Improve existing APIs
  > AJAX and Web 2.0
Scripting In The Web Tier

• Project Phobos – Add scripting to web tier
  > Complements client side AJAX JavaScript
  > https://phobos.dev.java.net

• JavaScript enabled pages:
  > Extra language choice for JSP pages
  > Lets you embed server-side (as well as client-side) JavaScript in pages

• Lightweight servlets in JavaScript
  > Invoke JavaScript on HTTP GETs and POSTs
Web Tier Futures – Even More AJAX

• We expect several usage styles to be popular
• Raw AJAX – handwritten JavaScript event handlers
  > Easy way to “sprinkle” AJAX on existing pages
• Widget Based AJAX – use prewritten JavaScript libraries for creating client user interface
  > Best approach for new applications
  > Can be encapsulated in JSF components
• Partial page refresh / partial page submit
  > See Project “Dynamic Faces”
  > https://jsf-extensions.dev.java.net
Web Tier Futures – Web 2.0

• I never thought I would see a buzzword with more hype than “AJAX” ...

• Driven by technology changes:
  > Enhanced UI expectations (AJAX, Flash, ...)  
  > Lightweight programming models
    > Scripting, simple APIs, iterative deployment

• And by larger scale considerations:
  > “Web as a platform” (REST, HTTP everywhere, software as a service)
  > “Read/write web” (social / community driven)
Web Tier Futures – Web 2.0

• At the same time, diverse technologies and platforms are very popular:
  > Example: Java Enterprise Edition
  > Example: Linux / Apache / MySQL / PHP (LAMP)
  > Example: Microsoft ASP.Net and successors

• What does success look like?
  > Being very good at target aspects of the problem
  > Embrace and interoperate with all technologies

• Thus, expanding the vision of the Java platform (beyond the Java language) makes sense
Summary
Summary – Themes To Remember

• Java Standard Edition 6:
  > Continued focus on core systemic benefits
  > Improved performance, features

• Java Enterprise Edition 5:
  > Substantial focus on ease of use

• Future directions – simplify, simplify, simplify!
  > Simpler APIs
  > Simpler languages
  > Interoperation with other platforms
“The Java ecosystem will continue to grow, to support my current and my future needs.”

Craig McClanahan
Java™ SE and EE Platforms: The High Order Bits

Craig R. McClanahan
Senior Staff Engineer
Sun Microsystems, Inc.